МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

**Звіт**

До лабораторної роботи №2

**Основи побудови об’єктно-орієнтованих додатків на Python**

Виконав:

ст. гр. ІТ-21сп

**Олександр БІЗЯЄВ**

Прийняв:

доц. каф. ІСМ

**Сергій ЩЕРБАК**

Львів ‒ 2023

**Лабораторна робота №2**

**Мета:** розробка консольного калькулятора в об’єктно орієнтованому стилі з використанням класів

**Хід роботи**

Завдання 1: Створення класу Calculator

Створіть клас Calculator, який буде служити основою для додатка калькулятора.

Завдання 2: Ініціалізація калькулятора

Реалізуйте метод \_\_init\_\_ у класі Calculator для ініціалізації необхідних атрибутів або змінних.

Завдання 3: Введення користувача

Перемістіть функціональність введення користувача в метод у межах класу Calculator. Метод повинен приймати введення для двох чисел і оператора.

Завдання 4: Перевірка оператора

Реалізуйте метод у класі Calculator, щоб перевірити, чи введений оператор є дійсним (тобто одним із +, -, \*, /). Відобразіть повідомлення про помилку, якщо він не є дійсним.

Завдання 5: Обчислення

Створіть метод у класі Calculator, який виконує обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення).

Завдання 6: Обробка помилок

Реалізуйте обробку помилок у межах класу Calculator для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідні повідомлення про помилку.

Завдання 7: Повторення обчислень

Додайте метод до класу Calculator, щоб запитати користувача, чи він хоче виконати ще одне обчислення. Якщо так, дозвольте йому ввести нові числа і оператор. Якщо ні, вийдіть з програми.

Завдання 8: Десяткові числа

Модифікуйте клас Calculator для обробки десяткових чисел (плаваюча кома) для більш точних обчислень.

Завдання 9: Додаткові операції

Розширте клас Calculator, щоб підтримувати додаткові операції, такі як піднесення до степеня (^), квадратний корінь (√) та залишок від ділення (%).

Завдання 10: Інтерфейс, зрозумілий для користувача

Покращте інтерфейс користувача у межах класу Calculator, надавши чіткі запити, повідомлення та форматування виводу для зручності читання.

Загальний код для виконання всіх цих завдань:

Файл **calculator.py**

class AdvancedCalculator(Calculator):  
  
 def \_calculation(self, first\_number, second\_number, operator):  
 if operator == 'sin':  
 return math.sin(math.radians(first\_number))  
 elif operator == 'cos':  
 return math.cos(math.radians(first\_number))  
 elif operator == 'tan':  
 return math.tan(math.radians(first\_number))  
 else:  
 return super().\_calculation(first\_number, second\_number, operator)

Файл **advanced\_calculator.py**

class AdvancedCalculator(Calculator):  
  
 def \_calculation(self, first\_number, second\_number, operator):  
 if operator == 'sin':  
 return math.sin(math.radians(first\_number))  
 elif operator == 'cos':  
 return math.cos(math.radians(first\_number))  
 elif operator == 'tan':  
 return math.tan(math.radians(first\_number))  
 else:  
 return super().\_calculation(first\_number, second\_number, operator)

Файл **variables.py**

first\_number\_input = "Please write number 1: "  
second\_number\_input = "Please write number 2: "  
operator\_input = "Please write operator(+, -, \*, /, %, ^, √, random, sin, cos, tan): "  
choose\_exit = "Do you want to perform another calculation? (Yes/No): "  
zero = "Error: The number can't be zero"  
sqrt = "Error: The number cannot be less than zero"  
random = "Error: first number must be less than second number"  
operator\_error = "Error: The operator entered is not valid. Enter the correct one."  
input\_number\_error = "Error: Incorrect number format entered. Please try again."

Файл errors.py

class Error:  
  
 def handle\_errors(self, first\_number, second\_number, operator):  
 if operator == '/':  
 if second\_number == 0:  
 try:  
 first\_number / second\_number  
 except ZeroDivisionError:  
 print(variables.zero)  
 return ZeroDivisionError  
 elif operator == '√':  
 try:  
 math.sqrt(first\_number)  
 except ValueError:  
 print(variables.sqrt)  
 return ValueError  
 elif operator == 'random':  
 try:  
 if first\_number > second\_number:  
 raise ValueError(variables.random)  
 except ValueError as e:  
 print(e)  
 return ValueError  
  
 def number\_input\_check(self, number):  
 try:  
 float(number)  
 except ValueError:  
 return ValueError()  
  
 def validate\_operator(self, operator):  
 try:  
 if operator not in ['+', '-', '\*', '/', '%', '^', '√', 'random', 'sin', 'cos', 'tan']:  
 raise ValueError()  
 except ValueError:  
 return ValueError()

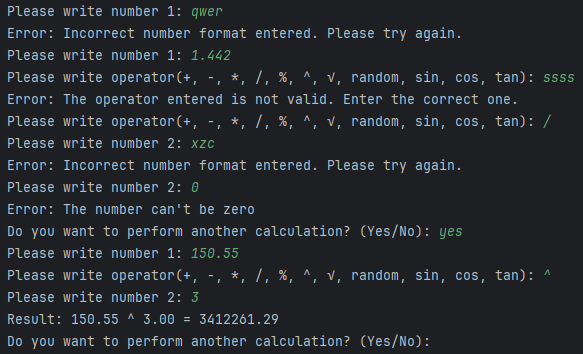
Файл **interface.py**

class Interface(AdvancedCalculator):  
  
 def \_choice(self):  
 return input(variables.choose\_exit).lower() != 'yes'  
  
 def run\_calculator(self):  
 error = Error()  
 while True:  
 first\_number, operator, second\_number = self.\_user\_input()  
  
 if not error.handle\_errors(first\_number, second\_number, operator):  
 self.result = self.\_calculation(first\_number, second\_number, operator)  
 result = self.result  
 if operator == '√':  
 print(f"Result: {first\_number:.2f} {operator} = {result:.2f}")  
 elif operator in ('sin', 'cos', 'tan'):  
 print(f"Result: {operator}({first\_number:.2f}°) = {result:.2f}")  
 elif operator == 'random':  
 print(f"Random number from {first\_number:.2f} to {second\_number:.2f} is {result:.2f}")  
 else:  
 print(f"Result: {first\_number:.2f} {operator} {second\_number:.2f} = {result:.2f}")  
  
 if self.\_choice():  
 break  
  
 def \_user\_input(self):  
 error = Error()  
  
 first\_number = self.get\_input(variables.first\_number\_input, error.number\_input\_check,  
 variables.input\_number\_error)  
  
 operator = ''  
 while error.validate\_operator(operator):  
 operator = input(variables.operator\_input)  
 if error.validate\_operator(operator):  
 print(variables.operator\_error)  
  
 second\_number = None  
 if operator not in ('√', 'sin', 'cos', 'tan'):  
 second\_number = self.get\_input(variables.second\_number\_input, error.number\_input\_check,  
 variables.input\_number\_error)  
  
 return first\_number, operator, second\_number  
  
 def get\_input(self, number, error\_check, error\_message):  
 while True:  
 value = input(number)  
 if not error\_check(value):  
 return float(value)  
 print(error\_message)

Файл **main.py**

if \_\_name\_\_ == "\_\_main\_\_":  
 interface = Interface()  
 interface.run\_calculator()

Результат виконання всіх завдань:



**Рис.1** Результат виконання

**Висновок:** Виконавши ці завдання, я розробив консольний калькулятора в об’єктно орієнтованому стилі з використанням класів